Покажувачи во c++

Секој податок во еден компјутерски систем, без разлика дали се работи за цел број, знак, текстуална низа или сложена структура, си има своја мемориска локација преку која може да биде пристапен. Во C++, преку креирање на променливи од соодветен тип, овозможена е работа со податоци без контрола на локацијата на која тие се сместени: всушност, во сите програми кои што ги напишавме досега, воопшто не се грижевме за локацијата на која се сместуваат податоците.

Покажувачите се посебен тип на променливи кои секогаш претставуваат адреса на (друга) мемориска локација т.е. покажуваат кон мемориската локација на дадената адреса. Покажувачот содржи позитивна целобројна вредност без предзнак, што се интерпретира како мемориска адреса (на која се чува друга променлива)

Променливите содржат вредности за податокот (директно референцирање)

Покажувачите содржат адреси на променливи (индиректно референцирање)

Покажувач е податочен типкој што чува (покажува кон) некоја мемориска локација. Оваа мемориска локација се чува преку нејзината адреса (некаков број).

Зошто служат покажувачите?

За брзо и ефикасно изминување на сложени податочни структури како низи и дрва, . . .
За ефикасно пренесување на сложени аргументи во функции.
Пренесување на низа, структура и слично
За пренесување на аргументи чии што вредности сакаме да останат во онаа состојба во која се наоѓаат по извршувањето на функцијата

Типовите на податоци што се користат во програмирањето можат да бидат:
- **статички** - се оние чија големина е однапред дефинирана (најчесто на почетокот на програмата). Тие се сместени на фиксни локации во меморијата
- **динамички** - се оние чија големина и/или структура се менува во текот на извршувањето на програмата. Тие не се сместуваат на фиксна локација во меморијата, туку на локација која во моментот на нивното креирање е слободна.

Динамичките типови на податоци можат да бидат:
- **со променлива големина** (низа битови, низа знаци, општа низа, множество, куп, ред и датотека)
- **со променлива структура** (листа, дрво, покажувач и објект).

При преведувањето на програмата потребно е преведувачот да ги знае сите променливи и нивниот тип. Променливите кои се создаваат за време на извршување на програмата, а потоа се бришат се нарекуваат **динамички променливи**. Динамичките променливи кои добиваат податоци од тип покажувач се наречени **променливи од тип покажувач или само покажувачи**. При креирањето, динамичките променливи се сместуваат во слободната внатрешна (динамичка) меморија, наречена heap-меморија.

Користењето на променливи од тип покажувач нуди две битни погодности во однос на меморијата:

1. Се проширува меморискиот простор што може да се користи за податоци во една програма
2. Со користење на покажувачките променливи во динамичката меморија, програмата може да се извршува со помала количина неопходна меморија. На пример, програмата може да поседува две многу сложени структури на податоци што не се користат истовремено. Ако овие две структури се декларираат како глобални променливи, тогаш тие се наоѓаат во сегментот за податоци и завземаат меморија за цело време на извршувањето на програмата, без оглед дали се користат во моментот или не. Но, ако се дефинирани преку покажувачи (динамички), тие се наоѓаат во динамичката меморија и ќе бидат избришани од неа по престанокот на нивното користење

Декларација на покажувачи

Покажувачот наместо податок содржи локација т.е. адреса на податокот што се наоѓа во динамичката меморија. Покажувачите се декларираат на следниот начин:
тип *име_на_покажувач;

каде што: **тип-** е типот на променлива на кои ќе покажува покажувачот, а " ***** " е знак за декларација на покажувач т.е покажувачка променлива.
Секој покажувач има тип. Типот се однесува на типот на променливата кон која тој покажува.
- При декларацијата за секој покажувач мора да се декларира неговиот податочен тип
- Вредноста на секој покажувач е позитивен цел број без предзнак (мемориска адреса), но, како се интерпретира вредноста која се наоѓа на оваа мемориска локација зависи од типот на покажувачот.

Пр. int *ip; // pokazivac na integer
double *dp; // pokazivac na double
float *fp; // pokazivac na float
char *ch // pokazivac na character
- разликуваме покажувач кон цел број, покажувач кон реален број, итн.
- може да се декларираат покажувачи од кој и да е податочен тип

Користење на покажувачи во C++

Основни операции кои се користат со покажувачи се:
1. Дефинирање променлива покажувач
2. Доделување адреса на пром. покажувач
3. Пристап до вредноста на адресата на која покажува покажувачот

Ова се прави со помош на операторот * кој ја враќа вредноста на пром. која се наоѓа на адресата што ја одредил неговиот операнд.
Пр.

```
#include <iostream>
using namespace std;
int main()
{
int var = 20;                    // deklaracija na prom
int *ip;                         // deklaracija na pokazuvac
ip = &var;                       // smestuvanje na adresata na var vo prom. Pokazuvac ip
cout << "Vrednosta na prom. var: ";
cout << var << endl;             // ja pecati adresata smestena vo pok. ip
cout << "adresata smestena vo pok. ip: ";
cout << ip << endl;              // pristap do vrednosta na adresata sto e dostapna vo pokazuvacot
cout << "Vrednost na *ip promenlivata ";
cout << *ip << endl;
return 0;
}
```

Во C++, покажувач се креира така што помеѓу типот на податок до кој се покажува и името на променливата која ќе служи како покажувач се поставува знакот '*' (ѕвезда). Мемориската локација на една променлива се добива со поставување на знакот '&' пред името на променливата чија адреса сакаме да ја дознаеме. Да разгледаме неколку примери:

Програма 1

```
#include <iostream>
using namespace std;
int main()
{
int a = 5; //promenliva 'a' so vrednost 5
int *b; //pokazhuvach kon podatok od tip int
b = &a; //b ja sodrzhi adresata na a ("b pokazhuva kon a")
/*
double *k;
k = &a; //GRESHKA: pogreshen tip na pokazhuvach (double)
*/
cout << a << endl; //pechati '5' (vrednosta na a)
cout << b << endl; //pechati '0x27ff44' (adresata na a)
return 0;
}
```

Најпрвин, важно е да се каже дека, при креирање на покажувачи (int *x), позицијата на која се наоѓа знакот '*' нема никакво влијание на извршувањето на програмата. Имено, знакот '*' може да се наоѓа веднаш по податочниот тип (int* x), помеѓу податочниот тип и името на покажувачот (int * x) или до името на покажувачот (int *x): сите наредби ќе имаат ист ефект.

Кога се креираат повеќе покажувачи од еден ист тип, треба да се внимава да се наведе знакот '*' за секој покажувач посебно:

Пр.

```
int a, b; //a e od tip 'int', b e od tip 'int'
int *a, *b; //a i b se pokazhuvachi kon podatok od tip 'int'
int *a, b; //a e pokazhuvach, b e obichna promenliva od tip 'int'
int a, *b; //a e od tip 'int', b e pokazhuvach
```

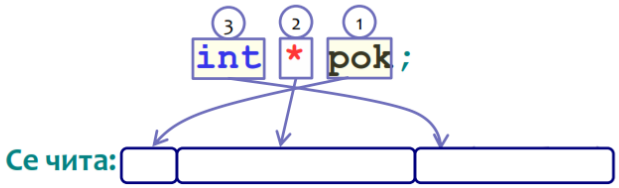
Често, кога користиме покажувачи, сакаме да знаеме која е вредноста која се чува на одредена мемориска локација - односно, која е вредноста на променливата до која покажува нашиот покажувач. Во C++, тоа го правиме на начин што пред името на покажувачот го поставуваме знакот '*' (истиот знак што го користевме за негово креирање). Тука е важно да се знае дека со знакот '*' всушност пристапуваме до податок на одредена мемориска локација и, не само што може да ја видиме неговата вредност, туку истата можеме и да ја промениме!

Програма 2

```
#include <iostream>
using namespace std;
int main()
{
int a = 5; //promenliva 'a' so vrednost 5
int *p; //pokazhuvach kon podatok od tip int
p = &a; //p ja sodrzhi adresata na a ("p pokazhuva kon a")
cout << a << endl; //pechati '5'
cout << *p << endl; //pechati '5'
*p = 3; //promena na vrednost
cout << p << endl; //pechati '0x27ff44' (adresa)
cout << *p << endl; //pechati '3' (vrednost)
```

```
cout << a << endl; //pechati '3'
return 0;
}
```

Читање на декларација на покажувач



Типот на променливата pok е int * (покажувач кон цел број)
double *myPtr1; Се чита: myPtr1 е покажувач кон double (реален број)

Операторот & (ampersand) е префикс оператор кој ја враќа мемориската адреса на која е сместена променливата

Оператор * (ѕвезда – asterisk) или оператор за дереференцирање -префикс оператор кој ја враќа содржината на мемориската локација чија адреса се наоѓа во променливата покажувач (pok)

```
int a,b;
int *pok;
a = 321;
pok = &a;
b=*pok;
```

Операторите * и &

Во делот на инструкциите (операторот за дереференцирање) ***** се чита како „**содржината на ...**“ или „**мемориската локација кон која покажува ...**“

```
*pok = *pok +1;
```

Операторот **&** се чита како „**адресата на ...**“

```
pok = &a;
```

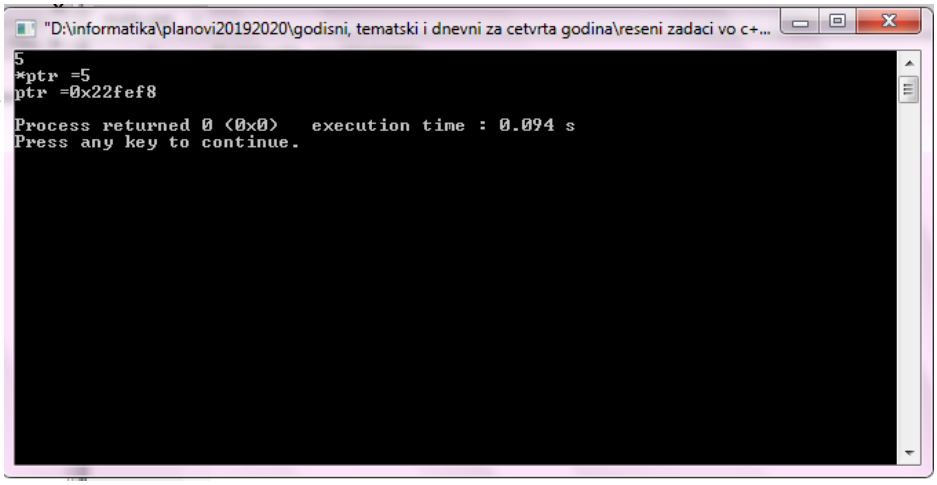
* се & инверзни и се поништуваат

Разликата помеѓу операторите '&' и '*' може да ја дефинираме со следниве два изрази:

- '&' се чита "адреса на" и ја враќа адресата на одреден податок. На пример, "&x" ја враќа адресата на x.
- '*' се чита "вредноста покажувана од" и ја враќа вредноста која се чува на одредена локација. На пример, "*p" служи за пристап до вредноста покажувана од покажувачот p.

Пр. Што ќе биде отпечатено?

```
#include <iostream>
using namespace std;
int main()
{
  int i=5,*ptr=&i;
cout<<i<<endl;
cout<<"*ptr ="<<*ptr<<endl;
cout<<"ptr ="<<ptr<<endl;
   return 0;
}
```



Адреса на i во меморијата

Доделување на вредност на покажувач

На нив може да се додели само адреса (вредност на друг покажувач од истиот тип)

Што значи **ptr + 1?**
Што означува **ptr - 1**?
Што означуваат **ptr*2** и **ptr/2**?

```
Пр. int x, *p, *q;
float *f;
void *v;
p=123;          не
p=&x;           да
q=p;            да
q=p+5;          да
v=p;            да
q=v;            не
f=q;            не
```

```
пр. int x=56, y;
int *p, *q;
float f, *r;
void *v;
p=123;              не
p=&x;               да
q=p;                да
q=p+5;              да
v=p;                да
q=v;                не
```

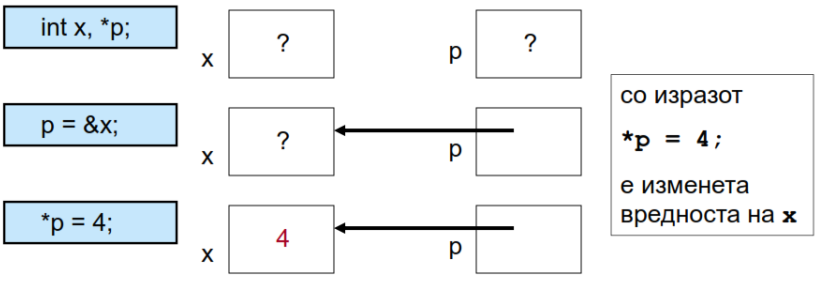
q=(int *)v;	да
*p=3.21;	не
*p=321;	да
r=&f;	да
*r=7.8;	да
*r=*q;	не
*v=456;	не

Програма 3

```
#include <iostream>
using namespace std;
int main()
{
int a = 5, b = 2;
int *pa = &a, *pb; //inicijalizacija
pb = &b; //pb pokazhuva na b
*pa = 3;
cout << a << " " << *pa << " " << *pb << endl; //pechati '3 3 2'
*pb = -1;
cout << b << " " << *pb << endl; //pechati '-1 -1'
*pa = *pb;
cout << a << " " << b << endl; //pechati '-1 -1'
pa = 0;
double c = 8.0123, d = 0.0000;
double *px;
px = &c;
(*px) += 2.0;
cout << (*px) << endl; //pechati '10.0123'
px = &d;
cout << (*px) << endl; //pechati '0'
return 0;
}
```

Менување на содржина со употреба на покажувач

```
int x, *p;
p = &x;
*p = 4;
```



Пример за користење на операторите & и *

```
#include <iostream>
using namespace std;
int main()
```

```
{
int x;
int *pok;
x=7;
pok=&x;
cout<<" Adresata na x e" << &x << "\n Vrednosta na pok e " << pok;
cout<<"Vrednosta na x e "<< x << "\n Vrednosta na *pok e "<< *pok;
return 0;
}
```

Забелешка: Операторите * и & се комплементни т.е. &*aPok=*&aPok

Пример за еден покажувач кога се користи да покажува кон различни променливи од ист тип:

```
#include <iostream>
using namespace std;
int main()
{
float c = 8.0123, d = 1.12345;
float *pok;
pok = &c;
(*pok) += 2.0;
cout << (*pok) << endl; //печати 10.0123
рок = &d;
(*pok)--;
cout << (*рок) << endl; // печати 0.12345
return 0;
}
```

Пример:

```
#include <iostream>
using namespace std;
int main()
{
int a = 5, b = 2;
int *pa = &a, *pb; // иницијализација на покажувач pa, и декларација на покажувач pb
pb = &b; //иницијализација на покажувач pb
*pa = 3;
cout << a << " " << *pa << " " << *pb << endl; // печати 3 3 2
*pb = -1;
cout << b << " " << *pb << endl; // печати -1 -1
*pa = *pb;
cout << a << " " << b << endl; // печати -1 -1
return 0;
}
```

Големината на еден покажувач зависи од архитектурата на компјутерскиот систем на кој се извршува програмата и од оперативниот систем: 32-битен компјутерски систем ќе користи 32-битни мемориски адреси (и, соодветно, покажувачите ќе зафаќаат 4 бајти податочен простор), додека 64-битен компјутерски систем ќе користи 64-битни мемориски адреси (и, соодветно, покажувачите ќе зафаќаат 8 бајти податочен простор). Ова на никој начин не влијае на однесувањето на покажувачите: тие и понатаму можат да се искористат за пристап до податокот на кој покажуваат - без разлика на неговата големина.

NULL покажувачи

Подобро е на покажувачот да му се додели вредност NULL во случај да не се знае точната адреса. NULL покажувач се декларира при декларацијата на пром. Покажувачот на кој му се доделува вредност NULL се вика **нулти покажувач.**
NULL покажувач е константа со вредност нула дефинирана во неколку стандардни c++ библиотеки, вклучувајќи ја и iostream.
Пр.

```
#include <iostream>
using namespace std;
int main()
{
int *ptr = NULL;
cout << "Vrednosta na ptr e " << ptr;
return 0;
}
```

Резултатот од извршувањето на прогрмата е:
Vrednosta na ptr e 0

Во повеќето оперативни системи не е дозволен пристап во меморијата на адреса 0 бидејќи таа е резервирана од о.с. Но мемориската адреса 0 означува дека покажувачот треба да се насочи на достапна мемориска локација. Но, ако покажувачот содржи вредност 0, се претпоставува дека не покажува на ништо.

Еден покажувач може да се искористи повеќе пати (во една иста програма) и притоа (во различни моменти) да покажува кон различни променливи и мемориски локации. Доколку сакаме експлицитно да изразиме дека одреден покажувач не покажува кон ништо, него може да му доделиме вредност 0 (т.н. NULL покажувач). Стандардот гарантира дека не постои податок во компјутерската меморија со адреса 0.

Алокација на меморија

Досега разгледавме неколку програми кои користеа низи за чување на повеќе податоци од еден ист тип. Но, едно нешто што беше константно во сите тие програми беше фактот што големината на низата ја определувавме однапред - како параметар при нејзиното креирање (int array[10]). Овој параметар, барем според C++ стандардот, мора да биде константен цел број.

Но, што доколку потребната големина не може да се специфицира пред самото извршување на програмата? На пример, што доколку сакаме да креираме низа од N елементи од тип студент, а бројот на студенти може да се открие единствено за време на извршување на програмата - како параметар кој го специфицира корисникот?

Решението на овој проблем е т.н. динамичко алоцирање на меморија. Во C++, тоа се прави со користење на операторите new (доколку сакаме да алоцираме простор за само еден елемент) или new[N] (доколку сакаме да алоцираме простор за N елементи). Резултатот од овие операции е покажувач до блокот меморија кој бил резервиран.

Многу е важно, по користењето на резервираната меморија (откако истата повеќе не е потребна), да се направи нејзино бришење. На тој начин, оперативниот систем потоа може да ја додели таа меморија на процесите кои навистина имаат потреба од неа. Во C++, ова се прави со користење на операторите delete (за еден елемент) и delete[] (за повеќе елементи).

Да разгледаме една програма која користи динамичко алоцирање на меморија:

Програма

```
#include <iostream>
using namespace std;
int main()
{
int *parr;
parr = new int;
*parr = 2;
cout << *parr << endl; //pechati '2'
delete parr; //MNOGU VAZHNO!!!
int N = 100;
parr = new int[N]; //niza od 100 elementi
parr[5] = 3;
cout << parr[5] << endl; //pechati '3'
delete [] parr; //MNOGU VAZHNO!!!
return 0;
}
```

Една од најчестите грешки при користењето на динамички алоцирана меморија е бришење на низа од N елементи со помош на операторот delete array - наместо delete[] array. Поради фактот што секој компјутерски систем има ограничено количество на меморија, понекогаш е можно да не успее динамичкото алоцирање на меморија (new int[N]). Притоа, доколку системот не успее да резервира доволно количество меморија, нашата програма ќе прекине со извршување и ќе јави грешка (runtime error).

Пр.

```
#include <iostream>
using namespace std;

void zgolemi (int* poc, int* kraj)
{
  int * tek = poc;
  while (tek != kraj)
   {
    ++(*tek);  // zgolemuva vrednost na koja pokazuva
    ++tek;     // zgolemuva pokazuvac
  }
}

void pecati ( int* poc, const int* kraj)
{
  const int * tek = poc;
  while (tek != kraj) {
    cout << *tek << '\n';
    ++tek;     //  zgolemuva pokazuvac
  }
}

int main ()
{
  int broj[] = {10,20,30};
  zgolemi (broj,broj+3);
```

```
  pecati (broj,broj+3);
  return 0;
}
```

Пр. Да се напише програма која ја печати мемориската адреса внесен цел број.

```
#include <iostream>
using namespace std;
int main()
{
int a, i;
cout<<"vnesi element "<<endl;
    cin>>a;
    cout<<"adresata na "<<a<<"  e "<<&a<<endl;
    return 0;
}
```

Пр. Да се напише програма која ги печати мемориските адреси на н внесени цели броеви.

```
#include <iostream>
using namespace std;
int main()
{
int a,n, i;
cout<<"vnesi broj na elementi "<<endl;
    cin>>n;
    for(i=1;i<=n;i++)
    {
       cout<<"vnesi broj "<<endl;
       cin>>a;
    cout<<"adresata na "<<a<<"  e "<<&a<<endl;
    }
    return 0;
}
```

Пр. Да се напише програма со која се внесуваат два броја и да се пресмета нивниот збир. Да се испечати вредноста на внесените броеви, добиениот збир и адрсите на користените променливи.

```
#include <iostream>
using namespace std;
int main()
{
        int a,b,c, *pa, *pb, *pc;
        pa = &a;
        pb = &b;
        pc = &c;
        cout<<" vnesi dva broja "<<endl;
        cin>>a>>b;
        *pa=a;
        *pb=b;
        c = a + b;
        *pc=c;
        cout<<*pa<<" e na adresa "<<pa<<endl;
```

```
        cout<<*pb<<" e na adresa "<<pb<<endl;
        cout<<" zbirot "<<*pc<<" e na adresa "<<pc<<endl;
         return 0;
}
```

Пр. Да се внесат три броја и да се испечати најголемиот од нив со користење на покажувачи.

```
#include <iostream>
using namespace std;
int main()
{
  int a, b, c, maxx,*pa, *pb, *pc, *pmax;
        pa = &a;
        pb = &b;
        pc = &c;
        pmax = &maxx;
        cout<<"vnesi tri broja"<<endl;
        cin>>a>>b>>c;
        *pa=a;
        *pb=b;
        *pc=c;
                *pmax = *pa;

        if (*pb > *pmax)
                *pmax = *pb;

        if (*pc > *pmax)
                *pmax = *pc;
cout<<" najgolem e "<<*pmax;
   return 0;
}
```

Пр. Да се внесат н знаци и да се испечати колку од нив се цифри со користење на покажувачи..

```
#include <iostream>
using namespace std;
int main()
{
  int i, br=0, *pbr=0,n;
        char zn, *pzn;
        pbr = &br;
        pzn = &zn;
        cout<<" vnesi broj na znaci"<<endl;
        cin>>n;
        for (i = 1; i <=n; i++)
      {
        cout<<" vnesi znak "<<endl;
      cin>>*pzn;
                if (*pzn >= '0' && *pzn <= '9')
                        (*pbr)++;
                }
```

```
cout<<"vneseni se "<<*pbr<<" broevi";

   return 0;
}
```

Пр.Да се внесат н броеви и да се испечати колку од внесените парни броеви се делат со 5, а колку непарни броеви се делат со 3 со користење на покажувачи..

```
#include <iostream>
using namespace std;
int main()
{
   int i, br, parni=0,n, neparni=0, *pbr, *ppar, *pnepar;

          pbr = &br;
          ppar = &parni;
          pnepar = &neparni;
cout<<" vnesi go n"<<endl;
cin>>n;
          for (i = 1; i<=n; i++)
       {
          cout<<"vnesi broj "<<endl;
          cin>>*pbr;

                    if (*pbr % 2 == 0 && *pbr % 5 == 0)
                              (*ppar)++;

                    if (*pbr % 2 == 1 && *pbr % 3 == 0)
                              (*pnepar)++;
          }
          cout<<"broj na parni broevi delivi so 5 e :"<<*ppar<<endl;
          cout<<"broj na neparni broevi delivi so 3 e :"<<*pnepar<<endl;
   return 0;
}
```

Пр. Да се внесат броеви се додека не се внесе 0. Да се испечати колку броеви се внесени чија вредност е меѓу 10 и 20. Да се иапечати најмалиот од тие броеви. Задачата да се реши со користење на покажувачи.

```
#include <iostream>
using namespace std;
int main()
{
   int i, br, minn=0, b=0, *pbr, *pmin, *pb=0;
          pbr = &br;
          pmin = &minn;
          pb = &b;
          i = 1;
cout<<"vnesi broj "<<endl;
cin>>*pbr;
          *pmin = *pbr;
```

```
        while (*pbr != 0)
    {
                if (*pbr >=10 && *pbr <= 20)
                        (*pb)++;

                if (*pbr < *pmin)
                        *pmin = *pbr;
                i++;
                cout<<"vnesi broj "<<endl;
    cin>>*pbr;
        }
                cout<<"megu 10 i 20 se "<<*pb<<endl;
                cout<<"najmal e "<<*pmin;
  return 0;
}
```

Пр. дали даден број е прост со покажувачи

```
#include <iostream>
using namespace std;
int main()
{
  int i, b=0, a, *pbr, *pa;
        pbr = &b;
        pa = &a;
        cout<<"vnesi broj"<<endl;
        cin>>*pa;
        for (i = 2; i <= *pa/2; i++)
                if(*pa % i == 0)
                        (*pbr)++;
        if (*pa == 1)
                cout<<"brojot 1 ne e ni prost ni slozen"<<endl;
        else
                if (*pbr == 0)
                        cout<<"Brojot e prost"<<endl;
                else
                        cout<<"Brojot ne e prost"<<endl;
  return 0;
}
```

Креирање на низа во динамичка меморија со помош на покажувач

Името на низата може да се идентификува со покажувачка константа на првиот член на низата. Значи, за низата a важи:

a ≡ &a[0]

Бидејќи елементите се наоѓаат на последователни мемориски локации,

a + 1 ≡ &a[1]

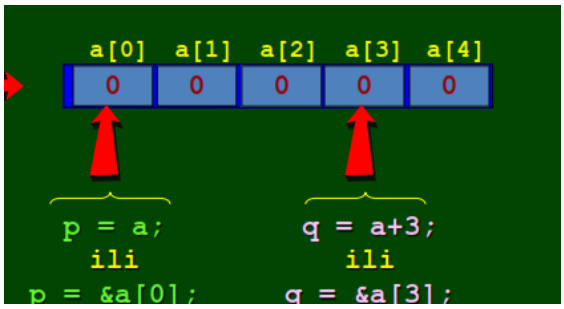
општо

a + i ≡ &a[i]

Кога на изразот ќе го примениме операторот за дереференцирање добиваме:

*(a + i) ≡ a[i]

Името на низата е адреса на почетниот елемент на низата и сама по себе е покажувач (статички покажувач)




```
int a[5]={0};
int *p, *q;
```

a е статичка променлива. Не е дозволено a=a+2 или a++.
p е динамичка променлива. Дозволено е p=p+2; или p++;
Пр.
Пристап и печатење на елементите на еднодимензионална низа со покажувач:

```
#include <iostream>
using namespace std;
int main()
{
   int x[5] = {1,2,3,4,5};
   int *p;
   p = x;
   cout << x[0]<<x[1]<<x[2]<< endl;
   cout << *p<< *(p+1)<<*(p+2)<< endl;
   return 0;
}
```

Адресна аритметика

Нека:
a- низа со елементи од тип т
v- израз од тип т
n -целоброен израз
p, q – се покажувачи од тип т (покажуваат на елементите од низата a)
Тогаш важат следниве правила:
a – е покажувач на почетокот на низата
&a[0] - е покажувач на почетокот на низата
*p – елемент од низата a на кој покажува p
*p=v – на елементот на кој покажува p му се доделува вредност v
++p – покажува на следниот елемент од низата
--p – покажува на претходниот елемент од низата
*++p – го зголемува p па пристапува до тој (следниот) елемент
*p++ - пристапува до елементот на кој покажува p, а потоа покажува на следниот елемент
p+n- покажува на n – ти нареден елемент од низата
*(p+n)=v – доделува вредност v на n-тиот нареден елемент во низата во однос на елементот на кој покажува p
q-p – одредува број на елементи сместени меѓу адресите на кои покажуваат q и p

Пр.
Да се внеси еднодимензионална низа со n броеви и да се пресмета збир на парните броеви со користење на покажувачи.

```
#include <iostream>
using namespace std;
int main()
{
   int a[100], i,n, z = 0, *pa;
         pa = &a[0];
  cout << "vnesi go n" << endl;
  cin>>n;
  for (i = 0; i < n; i++)
    {
                  cout<<"vnesi broj"<<endl;
      cin>>*(pa+i);
    }

         for (i = 0; i < n; i++)
      {
                  if (*(pa + i) % 2 == 0)
                        z+= *(pa + i);
      }
      cout<<"zbirot e "<<z;
   return 0;
}
```

Пр.Програма со која се испитува колку пати буквата А се јавува во низата.

```
# include <iostream>
using namespace std;
int main()
{
   char a[100];
   int n,i,b;
   cout<<"vnesi dolzina na nizata"<<endl;
   cin>>n;
   cout<<"vnesi gi elementite na nizata"<<endl;
   for (i=0; i<n; i++)
   cin>>*(a+i);
   b=0;
    for (i=0; i<n; i++)
    if (*(a+i) == 'A') b++;
    cout<<"bukvata A se javuva "<<b<<"  pati";
    return 0;
}
```

Пр.2. Програма со која се пресметува посебно збир на парни, посебно збир на непарни броеви во низа а со н елементи.

```
# include <iostream>
using namespace std;
int main()
```

```
{
   int a[100];
   int n,i,zp,zn;
   cout<<"vnesi dolzina na nizata"<<endl;
   cin>>n;
   cout<<"vnesi gi elementite na nizata"<<endl;
   for (i=0; i<n; i++)
   cin>>*(a+i);
   zp=0;
   zn=0;
    for (i=0; i<n; i++)
       if (*(a+i)%2 == 0) zp+=*(a+i);
         else zn+=*(a+i);

    cout<<"zbirot na parnite broevi vo nizata e "<<zp<<endl;
    cout<<"zbirot na neparnite broevi vo nizata e "<<zn<<endl;
    return 0;
}
```

Пр. Програма со која се наоѓа најголем елемент во низа а со н елементи и неговиот индекс

```
# include <iostream>
using namespace std;
int main()
{
   int a[100];
   int n,i,maxx, indeks;
   cout<<"vnesi dolzina na nizata"<<endl;
   cin>>n;
   cout<<"vnesi gi elementite na nizata"<<endl;
   for (i=0; i<n; i++)
   cin>>*(a+i);
  maxx=*(a); indeks=0;
    for (i=1; i<n; i++)
       if (*(a+i)>maxx)
       {
         maxx=*(a+i);
         indeks=i;
       }

    cout<<"najgolem e elementot "<<maxx<<endl;
    cout<<"negovata pozicija e "<<indeks<<endl;

    return 0;
}
```

Пр. Програма со која за низата броеви [ai]n, да се пресмета посебно збирот на позитивните и посебно збирот на негативните елементи.

```
#include<iostream>
using namespace std;
```

```
int main()
{
int a[50],i,n,zp,zn;
cout<<"Vnesi go brojot na elementi na nizata n=";
cin>>n;
cout<<"Vnesi gi elementite na nizata:"<<endl;
for(i=0;i<n;i++)
{
      cin >> *(a+i);
}
zp=0;
zn=0;
for(i=0;i<n;i++)
{
   if(*(a+i)>0)
      zp+=*(a+i);
   else
      zn+=*(a+i);
}
cout<<"Zbirot na pozitivnite elementi na nizata iznesuva: "<<zp<<endl;
cout<<"Zbirot na negativnite elementi na nizata iznesuva: "<<zn<<endl;

return 0;
}
```

Пр. Програма со која се пресметува колку броеви од н внесени во низа а се едноцифрени броеви.

```
# include <iostream>
using namespace std;
int main()
{
   int a[100];
   int n,i,b;
   cout<<"vnesi dolzina na nizata"<<endl;
   cin>>n;
   cout<<"vnesi gi elementite na nizata"<<endl;
   for (i=0; i<n; i++)
   cin>>*(a+i);
   b=0;
    for (i=0; i<n; i++)
    if (*(a+i) < 10) b++;
    cout<<"vo nizata ima  "<<b<<"  ednocifreni broevi";
    return 0;
}
```

Пр. Програма со која се формира нова низа чии елементи се збир на соодветните елементи од низите а и б.

```
# include <iostream>
using namespace std;
int main()
{
```

```
    int a[100],b[100],c[100];
    int n,i;
    cout<<"vnesi dolzina na nizite a i b"<<endl;
    cin>>n;
    cout<<"vnesi gi elementite na nizata a"<<endl;
    for (i=0; i<n; i++)
    cin>>*(a+i);
     cout<<"vnesi gi elementite na nizata b"<<endl;
    for (i=0; i<n; i++)
    cin>>*(b+i);
        for (i=0; i<n; i++)
           *(c+i)=*(a+i)+*(b+i);
    cout<<"nizata c e"<<endl;
    for (i=0; i<n; i++)
    cout<<*(c+i)<<"  ";
     return 0;
}
```

Пр.Да се напише програма со која се внесува низа a со н елементи. Потоа од низата a да се формира нова низа во која ќе бидат смесетени елементите помали од елементот што е на втора позиција во низата a.

```
#include <iostream>
using namespace std;
int main()
{
   int i,j,n;
   int a[100], b[100];
   cout<<"vnesi go n"<<endl;
   cin>>n;
   for(i=0;i<n;i++)
   {
      cin>>*(a+i);
   }
   j=0;
cout<<endl;
   for(i=0;i<n;i++)
   {
      if(*(a+i)<*(a+1))
      {
         *(b+j)=*(a+i);
         j++;
      }
   }
   cout<<endl;

    for(i=0;i<j;i++)
    {
       cout<<*(b+i)<<" ";
    }
    return 0;
```

}

Пр. Програма која печати на која позиција е првото појавување на даден број во еднодимензионална низа а со н елементи. Ако елементот не е во низата, да се даде соодветна порака.

```
#include <iostream>
using namespace std;

      void linearno(int *(a), int m, int i, int n)
      {   int t = 0;
         for (i = 0; i < n ; i++)
         {
            if (m == *(a+i))
            {
              t= 1;
               break;
            }
         }
         if (t == 1)
         {
            cout<<"Elementot e na pozicija   "<<i;
         }
         else
         {
            cout<<"Elementot ne e prisuten vo nizata.";
         }
      }

int  main()
{  int n, a[100];
   int i,  m, t = 0;
   cout<<"Vnesi go n   ";
   cin>>n;
      cout<<"Vnesi gi elementite"<<endl;
   for (i = 0; i < n; i++)
   {
      cin>> *(a+i);
   }
   cout<<"Vnesi go elementot sto go baras   ";
   cin>>m;

   linearno(a, m, i,n);
   return 0;
}
```

Пр. програма што ќе пресмета и печати колку од броевите во еднодимензионална низа со н елементи, се едноцифрени, колку се двоцифрени, а колку се трицифрени.

```
#include <iostream>
using namespace std;
```

```
    void linearno(int *(a), int n)
    {
       int i, e = 0, d=0, t=0;
       for (i = 0; i < n ; i++)
       {
          if (*(a+i)>=0 && *(a+i)<=9) e++;
          if (*(a+i)>=10 && *(a+i)<=99) d++;
          if (*(a+i)>=100 && *(a+i)<=999) t++;
       }
             cout<<"Ednocifreni se   "<<e<<endl;
             cout<<"Dvocifreni se   "<<d<<endl;
             cout<<"Tricifreni se   "<<t<<endl;
    }

int  main()
{  int n,i, a[100];
   cout<<"Vnesi go n   ";
   cin>>n;
      cout<<"Vnesi gi elementite"<<endl;
   for (i = 0; i < n; i++)
   {
      cin>> *(a+i);
   }
     linearno(a,n);
   return 0;
}
```

Пр. Програма со која од дадена еднодимензионална низа со н елементи креира други две низи, едната составена од парните, а другата од непарните елементи на низата а.

```
#include <iostream>
using namespace std;

    void baraj(int *a, int n, int *j1, int *j2, int *b, int *c)
    {
       int i,p, j=0, k=0;
       for (i = 0; i < n ; i++)
       {
          if (*(a+i)%2==0)
          {
            *(b+j)=*(a+i);
            j++;

          }
          else
          {
             *(c+k)=*(a+i);
            k++;
          }
```

```
        }
          *j1=j;
          *j2=k;
      }

int  main()
{  int n,i, j,k,a[100],b[100],c[100],j1, j2;
   cout<<"Vnesi go n   ";
   cin>>n;
      cout<<"Vnesi gi elementite"<<endl;
   for (i = 0; i < n; i++)
   {
      cin>> *(a+i);
   }
     baraj(a,n, &j1, &j2, b,c);
    cout<<"Elementi na nizata b se  "<<endl;
        for(i=0;i<j1;i++)

            cout<<*(b+i)<<"  ";
        cout<<endl;
        cout<<"Elementi na nizata c se  "<<endl;
        for(i=0;i<j2;i++)
           cout<<*(c+i)<<"  ";
        cout<<endl;
   return 0;
}
```

Пр. Програма со која од дадена еднодимензионална низа со н елементи креира други две низи, едната составена од елементите од првата десетка, а другата од останатите елементи на низата а.

```
#include <iostream>
using namespace std;

      void kreiraj(int *a, int n, int *j1, int *j2, int *b, int *c)
      {
         int i,p, j=0, k=0;
         for (i = 0; i < n ; i++)
         {
            if (*(a+i)>=0 && *(a+i)<=9)
            {
              *(b+j)=*(a+i);
              j++;

            }
            else
            {
               *(c+k)=*(a+i);
              k++;
            }
         }
```

```
            *j1=j;
            *j2=k;
        }

int  main()
{  int n,i, j,k,a[100],b[100],c[100],j1, j2;
   cout<<"Vnesi go n   ";
   cin>>n;
      cout<<"Vnesi gi elementite"<<endl;
   for (i = 0; i < n; i++)
   {
      cin>> *(a+i);
   }
     kreiraj(a,n, &j1, &j2, b,c);
    cout<<"Elementi na nizata b se  "<<endl;
         for(i=0;i<j1;i++)

            cout<<*(b+i)<<"  ";
         cout<<endl;
         cout<<"Elementi na nizata c se  "<<endl;
         for(i=0;i<j2;i++)
            cout<<*(c+i)<<"  ";
         cout<<endl;
   return 0;
}
```

Пр. Да се напише програма со која се подредуваат елементите од низа а со н елементи во растечки редослед и да се отпечати новодобиената низа.

```
#include <iostream>
using namespace std;

       void podredi(int *a, int n)
       {
          int i,p, j=0, k=0;
          for (i = 0; i < n-1 ; i++)
          {
             for(j=i+1;j<n; j++)
             {
                if(*(a+i)>*(a+j))
                   {
                   p=*(a+i);
                   *(a+i)=*(a+j);
                   *(a+j)=p;
                   }
             }
          }

       }
```

```
void  pecati(int *a, int n)
{
int i;
    for(i=0;i<n;i++)
    {
       cout<<*(a+i)<<" ";
    }
}


int  main()
{  int n,i,a[100];
    cout<<"Vnesi go n   ";
    cin>>n;
       cout<<"Vnesi gi elementite"<<endl;
    for (i = 0; i < n; i++)
    {
       cin>> *(a+i);
    }
      podredi(a,n);
     cout<<"Elementi na nizata a se  "<<endl;

pecati(a,n);

    return 0;
}
```

Пр.Да се напише програма со која се внесува низа a со н елементи. Потоа од низата a да се формира нова низа во која ќе бидат смесетени елементите помали од елементот што е на втора позиција во низата a.

```
#include <iostream>
using namespace std;
void vnesi(int *a, int n)
{
    int i;
   for(i=0;i<n;i++)
   {
      cin>>*(a+i);
   }
}
void  nova(int *a, int n, int *b,  int *j1)
{
    int i, j=0;
    *j1=0;
   for(i=0;i<n;i++)
   {
      if(*(a+i)<*(a+1))
      {
         *(b+j)=*(a+i);
```

```
        j++;
        cout<<"J="<<j<<"  ";
      }
   }
   cout<<endl;
   *j1=j;
  }

void  pecati(int *b, int &j1)
{
int i;
    for(i=0;i<j1;i++)
    {
       cout<<*(b+i)<<" ";
    }
}

int main()
{
   int i,j,n, j1;
   int a[100], b[100];
   cout<<"vnesi go n"<<endl;
   cin>>n;
vnesi(a,n);
   j=0;
cout<<endl;
nova(a, n, b, &j1);
//cout<<"sozdadena e b so "<<j1 <<"elementi"<<endl;
pecati(b, j1);
    return 0;
}
```